MEMORY HUB AND ACCESS METHOD HAVING INTERNAL PREFETCH BUFFERS

TECHNICAL FIELD

This invention relates to computer systems, and, more particularly, to a computer system having a memory hub coupling several memory devices to a processor or other memory access device.

BACKGROUND OF THE INVENTION

Computer systems use memory devices, such as dynamic random access memory ("DRAM") devices, to store data that are accessed by a processor. These memory devices are normally used as system memory in a computer system. In a typical computer system, the processor communicates with the system memory through a processor bus and a memory controller. The processor issues a memory request, which includes a memory command, such as a read command, and an address designating the location from which data or instructions are to be read. The memory controller uses the command and address to generate appropriate command signals as well as row and column addresses, which are applied to the system memory. In response to the commands and addresses, data are transferred between the system memory and the processor. The memory controller is often part of a system controller, which also includes bus bridge circuitry for coupling the processor bus to an expansion bus, such as a PCI bus.

Although the operating speed of memory devices has continuously increased, this increase in operating speed has not kept pace with increases in the operating speed of processors. Even slower has been the increase in operating speed of memory controllers coupling processors to memory devices. The relatively slow speed of memory controllers and memory devices limits the data bandwidth between the processor and the memory devices.

In addition to the limited bandwidth between processors and memory devices, the performance of computer systems is also limited by latency problems that

increase the time required to read data from system memory devices. More specifically, when a memory device read command is coupled to a system memory device, such as a synchronous DRAM ("SDRAM") device, the read data are output from the SDRAM device only after a delay of several clock periods. Therefore, although SDRAM devices can synchronously output burst data at a high data rate, the delay in initially providing the data can significantly slow the operating speed of a computer system using such SDRAM devices.

One approach to alleviating the memory latency problem is to use multiple memory devices coupled to the processor through a memory hub. In a memory hub architecture, a system controller or memory controller is coupled to several memory modules, each of which includes a memory hub coupled to several memory devices. The memory hub efficiently routes memory requests and responses between the controller and the memory devices. Computer systems employing this architecture can have a higher bandwidth because a processor can access one memory device while another memory device is responding to a prior memory access. For example, the processor can output write data to one of the memory devices in the system while another memory device in the system is preparing to provide read data to the processor. Although computer systems using memory hubs may provide superior performance, they nevertheless often fail to operate at optimum speed for several reasons. For example, even though memory hubs can provide computer systems with a greater memory bandwidth, they still suffer from latency problems of the type described above. More specifically, although the processor may communicate with one memory device while another memory device is preparing to transfer data, it is sometimes necessary to receive data from one memory device before the data from another memory device can be used. In the event data must be received from one memory device before data received from another memory device can be used, the latency problem continues to slow the operating speed of such computer systems.

One technique that has been used to reduce latency in memory devices is to prefetch data, i.e., read data from system memory before the data are requested by a program being executed. Generally the data that are to be prefetched are selected based

on a pattern of previously fetched data. The pattern may be as simple as a sequence of addresses from which data are fetched so that data can be fetched from subsequent addresses in the sequence before the data are needed by the program being executed. The pattern, which is known as a "stride," may, of course, be more complex.

Although data prefetching can reduce memory access latencies in conventional computer systems, prefetching of data has not been effectively used in a manner that provides optimum performance in computer systems using memory hubs. In particular, the vast amount of data that can be addressed in a computer system having several memory hubs makes it difficult to accurately predict which data will be subsequently needed. Furthermore, even if the data that will be required can be correctly anticipated, it can be unduly time consuming to couple the data from memory devices in a memory module, and through a memory hub in the memory module to a prefetch buffer in the system controller or memory controller. The need to couple the data from the memory module to the prefetch buffer can also reduce the memory bandwidth of the system if the data are being prefetched at a time when normal memory accesses are being attempted.

There is therefore a need for a computer architecture that provides the advantages of a memory hub architecture and also minimize the latency problems common in such systems, thereby providing memory devices with high bandwidth and low latency.

SUMMARY OF THE INVENTION

A memory module that may be used in a computer system includes a plurality of memory devices coupled to a memory hub. The memory hub includes a link interface receiving memory requests for access to memory cells in at least one of the memory devices. A memory device interface couples memory requests to the memory devices and receives read data responsive to at least some of the memory requests. A history logic unit included in the memory hub receives memory requests from the link interface and predicts on the basis of the memory requests the addresses in the memory devices that are likely to be accessed. The history logic unit then generates prefetching

suggestions indicative of the predicted addresses. The memory hub also includes a memory sequencer that couples memory requests to the memory device interface responsive to memory requests received from the link interface. The memory sequencer also generates and couples prefetching requests to the memory device interface responsive to prefetching suggestions received from the history logic unit. A prefetch buffer included in the memory hub receives and stores read data from memory cells being accessed responsive to the prefetching requests. Finally, a data read control unit included in the memory hub determines from a read memory request received from the link interface if the read data are stored in the prefetch buffer. If the read data are stored in the prefetch buffer, the read data are read from the prefetch buffer. If the read data are not stored in the prefetch buffer, the read data are read from the memory devices.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a computer system according to one example of the invention in which a memory hub is included in each of a plurality of memory modules.

Figure 2 is a block diagram of a memory hub used in the computer system of Figure 1, which contains a prefetch buffer according to one example of the invention.

DETAILED DESCRIPTION OF THE INVENTION

A computer system 100 according to one example of the invention is shown in Figure 1. The computer system 100 includes a processor 104 for performing various computing functions, such as executing specific software to perform specific calculations or tasks. The processor 104 includes a processor bus 106 that normally includes an address bus, a control bus, and a data bus. The processor bus 106 is typically coupled to cache memory 108, which, as previously mentioned, is usually static random access memory ("SRAM"). Finally, the processor bus 106 is coupled to a system controller 110, which is also sometimes referred to as a "North Bridge" or "memory controller."

The system controller 110 serves as a communications path to the processor 104 for a variety of other components. More specifically, the system controller 110 includes a graphics port that is typically coupled to a graphics controller 112, which is, in turn, coupled to a video terminal 114. The system controller 110 is also coupled to one or more input devices 118, such as a keyboard or a mouse, to allow an operator to interface with the computer system 100. Typically, the computer system 100 also includes one or more output devices 120, such as a printer, coupled to the processor 104 through the system controller 110. One or more data storage devices 124 are also typically coupled to the processor 104 through the system controller 110 to allow the processor 104 to store data or retrieve data from internal or external storage media (not shown). Examples of typical storage devices 124 include hard and floppy disks, tape cassettes, and compact disk read-only memories (CD-ROMs).

The system controller 110 is coupled to several memory modules 130a,b...n, which serve as system memory for the computer system 100. The memory modules 130 are preferably coupled to the system controller 110 through a high-speed link 134, which may be an optical or electrical communication path or some other type of communications path. In the event the high-speed link 134 is implemented as an optical communication path, the optical communication path may be in the form of one or more optical fibers, for example. In such case, the system controller 110 and the memory modules will include an optical input/output port or separate input and output ports coupled to the optical communication path. The memory modules 130 are shown coupled to the system controller 110 in a multi-drop arrangement in which the single high-speed link 134 is coupled to all of the memory modules 130. However, it will be understood that other topologies may also be used, such as a point-to-point coupling arrangement in which a separate high-speed link (not shown) is used to couple each of the memory modules 130 to the system controller 110. A switching topology may also be used in which the system controller 110 is selectively coupled to each of the memory modules 130 through a switch (not shown). Other topologies that may be used will be apparent to one skilled in the art.

Each of the memory modules 130 includes a memory hub 140 for controlling access to 8 memory devices 148, which, in the example illustrated in Figure 2, are synchronous dynamic random access memory ("SDRAM") devices. However, a fewer or greater number of memory devices 148 may be used, and memory devices other than SDRAM devices may, of course, also be used. The memory hub 140 is coupled to each of the system memory devices 148 through a bus system 150, which normally includes a control bus, an address bus and a data bus.

One example of the memory hub 140 of Figure 1 is shown in Figure 2. The memory hub 140 includes a link interface 152 that is coupled to the high-speed link 134. The nature of the link interface 152 will depend upon the characteristics of the high-speed link 134. For example, in the event the high-speed link 134 is implemented using an optical communications path, the link interface 152 will include an optical input/output port or separate input and output ports and will convert optical signals received through the optical communications path into electrical signals and electrical signals into optical signals that are transmitted to the optical communications path. In any case, the link interface 152 may include a variety of conventional interface circuitry such as, for example, a first-in, first-out buffer (not shown), for receiving and storing memory requests as they are received through the high-speed link 134. The memory requests can then be stored in the link interface until they can be processed by the memory hub 140.

A memory request received by the link interface 152 is processed by first transferring the request to a memory sequencer 160. The memory sequencer 160 converts the memory requests from the format output from the system controller 110 (Figure 1) into a memory request having a format that can be used by the memory devices 148. These re-formatted request signals will normally include memory command signals, which are derived from memory commands contained in the memory request received by the memory hub 140, and row and column address signals, which are derived from an address contained in the memory request received by the memory hub 140. In the event the memory request is a write memory request, the re-formatted request signals will normally include write data signals which are derived from write

data contained in the memory request received by the memory hub 140. For example, where the memory devices 148 are conventional DRAM devices, the memory sequencer 160 will output row address signals, a row address strobe ("RAS") signal, an active low write/active high read signal ("W*/R"), column address signals and a column address strobe ("CAS") signal. The re-formatted memory requests are preferably output from the sequencer 160 in the order they will be used by the memory devices 148.

The memory sequencer 160 applies the re-formatted memory requests to a memory device interface 166. The nature of the memory device interface 166 will again depend upon the characteristics of the memory devices 148. In any case, the memory device interface 166, like the link interface 152, may include a FIFO buffer (not shown), for receiving and storing one or more memory requests as they are received from the link interface 152. The memory request can be stored in the FIFO buffer until they can be processed by the memory devices 148. Alternatively, the memory device interface can simply pass the memory requests to the memory devices 148.

In the event the memory device interface 166 stores several memory requests until they can be processed by the memory devices 148, the memory device interface 166 may re-order the memory requests so that they are applied to the memory devices 148 in some other order. For example, the memory requests may be stored in the interface 166 in a manner that causes one type of request, e.g., read requests, to be processed before other types of requests, e.g., write requests.

The memory requests are described above as being received by the memory hub 140 in a format that is different from the format that the memory requests are applied to the memory devices 148. However, the system controller 110 may instead re-format memory requests from the processor 104 (Figure 1) to a format that can be used by the memory devices 148. In such case, it is not necessary for the sequencer 160 to re-format the memory request. Instead, the sequencer 160 simply schedules the re-formatted memory request signals in the order needed for use by the memory devices 148. The memory request signals for one or more memory requests are

then transferred to the memory device interface 166 so they can subsequently be applied to the memory devices 148.

As previously explained, one of the disadvantages of using memory hubs is the increased latency they can sometimes create. As also previously explained, prefetch approaches that are traditionally used to reduce memory read latency are not well suited to a memory system using memory hubs. In contrast, the memory hub 140 shown in Figure 2 provides relatively low memory read latency by including a prefetch system 170 in the memory hub 140 that correctly anticipates which data will be needed during execution of a program, and then prefetches those data and stores them in one or more buffers that are part of the prefetch system 170. The prefetch system 170 includes several prefetch buffers 176, the number of which can be made variable depending upon operating conditions, as explained in greater detail below. The prefetch buffers 176 receive prefetched data from the memory device interface 166. The data are stored in the prefetch buffers 176 so that they will be available for a subsequent memory access. The data are then coupled through a multiplexer 178 to the link interface 152.

The prefetch system 170 also includes history logic 180 that receives the memory requests from the link interface 152. The history logic 180 analyzes the memory request using conventional algorithms to detect a pattern or stride from which future memory requests can be predicted. The history logic 180 couples prefetching suggestions to the memory sequencer 160, which then generates corresponding prefetching requests to read the suggested data. The memory sequencer 160 preferably prefetches data from the memory devices 148 for storage in the prefetch buffers 176 when the memory hub 140 is not busy responding to memory requests from the system controller 110. More specifically, when the sequencer 160 is not busy servicing memory requests from the link interface 152, the sequencer 160 generates the prefetch requests based on the prefetching suggestions, which are applied to the memory device interface 166. Prefetch data read from the memory devices 148 responsive to the prefetching requests are stored in the prefetch buffers 176. The prefetch data are stored in the prefetch buffers 176 along with identifying information, such as the address from

which the data were read to allow the correct data to be subsequently read from the memory devices 148.

Although data may be prefetched from any address in the memory devices 148, the data are preferably prefetched only from rows in the memory devices 148 that are currently active or "open" so that the prefetching will not require a row of memory cells in the memory devices 148 to be precharged.

The history logic 180 may also detect the existence of several strides from which different sets of memory requests can be predicted. For example, the history logic 180 may detect a first stride containing addresses 100, 101, 102..., a second stride containing addresses 305, 405, 505..., and a third stride containing addresses 300, 304, 308.... Data being read responsive to memory requests that are in different strides are preferably stored in different sections of the prefetch buffers 176. The data read from addresses 100, 101, 102... in the first stride are preferably stored in a first section of the prefetch buffers 176, data read from addresses 305, 405, 505... in the second stride are preferably stored in a second section of the prefetch buffers 176, data read from addresses 300, 304, 308...a third stride are preferably stored in a third section of the prefetch buffers 176, etc. Therefore, the history logic 180 also preferably determines the number of strides in existence and enables or creates respective sections of the prefetch buffers 176 to store the data read from the addresses that are in the corresponding stride. The sections of the prefetch buffers 176 may be enabled or created using a variety of conventional techniques. For example, the prefetch buffers 176 may be implemented as a single static random access memory ("SRAM") device that is partitioned into a number of sections corresponding to the number of strides in existence. The prefetch buffers 176 may also be separate registers or memory devices that are enabled as they are needed to store data from a respective stride. Other means of dividing the prefetch buffers 176 into different sections will be apparent to one skilled in the art. For example, in addition to adjusting the number of sections created in the prefetch buffers 176, the history logic 180 may adjust the size of each prefetch buffer section to match the amount of prefetch data in each stride.

The history logic 180 may also selectively enable or disable prefetching depending on whether or not a stride is detected by the history logic 180. However, prefetching may also be enabled all of the time. If the memory requests applied to the history logic 180 have very little locality, i.e., they are for addresses in different rows of memory or are somewhat random, it may be desirable to disable prefetching. If, however, the memory requests applied to the history logic 180 have good locality, the history logic 180 may enable prefetching. Alternatively, the history logic 180 may enable or disable prefetching based on the percentage of memory requests that result in reading the requested data from the prefetch buffers 176 rather than from the memory devices 148.

When a memory module 130 containing a memory hub 140 receives a read memory request, it first determines whether or not the data or instruction called for by the request is stored in the prefetch buffers 176. This determination is made by coupling the memory request to tag logic 186. The tag logic 186 receives prefetch addresses from the history logic 180 corresponding to each prefetch suggestion. Alternatively, the tag logic 186 could receive prefetch addresses from the memory sequencer 160 corresponding to each prefetch request coupled to the memory device interface 166. Other means could also be used to allow the tag logic 186 to determine if data called for by a memory read request are stored in the prefetch buffer 176. In any case, the tag logic 186 stores the prefetch addresses to provide a record of the data that have been stored in the prefetch buffers 176. Using conventional techniques, the tag logic 186 compares the address in each memory request received from the link interface 152 with the prefetch addresses stored in the tag logic 186 to determine if the data called for by the memory request are stored in the prefetch buffers 176. If the tag logic 186 determines the data are not stored in the prefetch buffers 176, it couples a low HIT/MISS* signal to the memory sequencer 160.

The memory sequencer 160 responds to a low HIT/MISS* signal by coupling the memory request received from the link interface 152 to the memory device interface 166 for coupling to the memory devices 148. The data called for by the memory request are then read from the memory devices 148 and coupled to the memory

device interface 166. The low HIT/MISS* signal is also applied to the multiplexer 178, thereby causing the multiplexer 178 to couple the read data from the memory device interface 166 to the link interface 152. The time required for all of these events to occur responsive to a memory request can be considerable, and may result in a considerable read latency. It is for this reason that data prefetching is desirable.

If the Tag Logic 186 determines the data called for by a memory request are stored in the prefetch buffers 176, it couples a high HIT/MISS* signal to the memory sequencer 160. The sequencer 160 then couples the memory request received from the link interface 152 to the prefetch buffers 176 rather than to the memory device interface 166, as was the case for a low HIT/MISS* signal. The data called for by the memory request are then read from the prefetched buffers 176 and applied to the multiplexer 178. The high HIT/MISS* signal causes the multiplexer 178 to couple the read data from the prefetch buffers to the link interface 152.

From the foregoing it will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various modifications may be made without deviating from the spirit and scope of the invention. Accordingly, the invention is not limited except as by the appended claims.